INFORMATION AND CONTROL 20, 114–124 (1972)

Elimination of Static and Dynamic Hazards for Multiple Input Changes in Combinational Switching Circuits*

JON G. BREDESON AND PAUL T. HULINA

Department of Electrical Engineering, The Pennsylvania State University, University Park, Pennsylvania 16802

This paper deals with hazards on outputs of combinational switching circuits for multiple input changes. Certain types of function hazards are defined and are shown to be impossible to eliminate with any logic realization. Also, an interrelation between static and dynamic function hazards is established. Hazards due to delays in the logic are defined and a method of elimination is given for both the static and dynamic case.

I. INTRODUCTION

Spurious pulses which may occur on the output of a combinational or sequential switching circuit during an input change are called hazards for the input change. Unless these hazards are eliminated, improper sequential operation may occur. Procedures for detecting and removing hazards in combinational networks for single input variables changes have been given by Huffman (1957) and McCluskey (1962).

Eichelberger (1964) has defined two types of hazards, logic and function, which are associated with multiple input changes when the initial and final outputs are identical. Function hazards were shown to be inherent in the Boolean function and could not be eliminated. Logic hazards can be eliminated by using all prime implicants in a sum-of-products form.

The elimination of hazards for multiple input changes will be treated in this paper when the initial and final outputs are different. It will be shown that similar logic and function hazards occur for this case.

* This paper was presented in part at the 11th Annual Symposium on Switching and Automata Theory in Santa Monica, California, October 28–30, 1970.

II. Function Hazards

Before a hazard-free realization can be proposed, some definitions are in order. An input state for any function will be a n-tuple $(x_1, x_2, ..., x_n)$. An arbitrary input state will be represented by $A = (a_1, a_2, ..., a_n)$, where a_i represents a logic value for input x_i.

Definition 1. An input state A is said to be in the *subcube* of input states $A_1, A_2, ..., A_m$ [written $A \in (A_1, A_2, ..., A_m)$] if and only if for all x_i variables which have the same logic value for $A_1, A_2, ..., A_m$; then the x_i variable has the same logic value for A.

As an example of the definition consider the following input states:

$$A_1 = (1, 0, 0, 1), \qquad A_2 = (0, 0, 0, 1),$$
$$A_3 = (0, 0, 0, 0), \qquad A = (1, 0, 0, 0).$$

By the definition

$$A \in (A_1, A_2, A_3), \qquad A \in (A_1, A_3)$$

but

$$A \notin (A_1, A_2).$$

The following definition is equivalent to Eichelberger's (1964) definition of a function hazard.

Definition 2. A Boolean function f contains a *static function hazard* for the input change A to C if and only if

(1) $f(A) = f(C)$.

(2) There exists at least one input state $B \in (A, C)$ such that $f(B) \neq f(A)$.

Definition 3. A Boolean function f contains a *dynamic function hazard* for the input change A to D if and only if

(1) $f(A) \neq f(D)$.

(2) There exists at least one pair of inputs states B and C such that

a. $B \in (A, D)$ and $C \in (B, D)$ and

b. $f(B) = f(D)$ and $f(C) = f(A)$.

The function f in Fig. 1 has a static function hazard for input change A to C, i.e., $f(A) = f(C) = 0, f(B) = 1$, and $B \in (A, C)$. A dynamic function

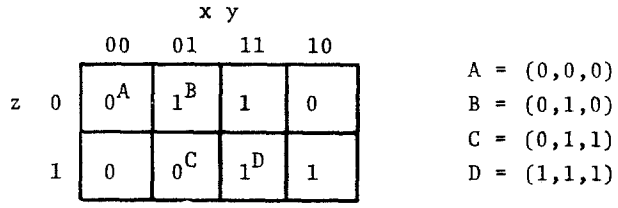


FIG. 1. Example of function hazards.

hazard exists for input change A to D since $f(A) = 0, f(D) = 1$, $B \in (A, D)$, $C \in (B, D), f(A) = f(C) = 0$, and $f(B) = f(D) = 1$.

It should be apparent that if a function hazard exists (either static or dynamic), a sequence of input changes can exist where a function may have a multiple (more than one) output change. Then it is possible that delays in the network realizing a function with function hazards could cause a multiple change in the output of the network. Thus the following theorem:

THEOREM 1. *If a Boolean function f contains a static or dynamic function hazard for an input change A to B, it is impossible to realize a network for f which will eliminate possible multiple output changes for input change A to B.*

There is a definite relationship between static and dynamic function hazards which is given by the following theorem:

THEOREM 2. *A Boolean function f contains a dynamic function hazard for the input change A to D if and only if there exists a static function hazard for the input change B to D for some $B \in (A, D)$, where $f(A) \neq f(D)$.*

Proof. Assume a static function hazard exists for the input change B to D, where $B \in (A, D)$ and $f(A) \neq f(D)$. Then, by the definition of a static function hazard, there exists an input state C such that $C \in (B, D)$ and $f(C) \neq f(B)$. Since $f(B) = f(D)$ and $f(C) = f(A)$, all conditions are satisfied for a dynamic function hazard for input change A to D.

Assume a dynamic function hazard exists for the input change A to D. This implies there exists a pair of input states B and C such that $B \in (A, D)$, $C \in (B, D)$, $f(B) = f(D)$ and $f(C) \neq f(B)$. A static function hazard then exists for input change B to D such that $B \in (A, D)$. Hence, the theorem.

Theorem 3 also relates static and dynamic function hazards.

THEOREM 3. *If a Boolean function f contains a dynamic function hazard for the input change A to D, then f contains a static function hazard for each*

input change A to C and B to D, where $B \in (A, D)$, $C \in (B, D)$, $f(B) = f(D)$ and $f(A) = f(C)$. Furthermore, $f(A) = f(C) = 0$ and $f(B) = f(D) = 1$ or vice versa.

Proof. Assume a dynamic function hazard exists for input change A to D. Therefore, there exist input states B and C such that $B \in (A, D)$, $C \in (B, D)$ and $f(A) = f(C) \neq f(B) = f(D)$. Obviously, a static function hazard exists for input change B to D since $C \in (B, D)$ and $f(C) \neq f(B) = f(D)$. If it can be shown $B \in (A, C)$, then a static function hazard exists for input change A to C. Assume $B \notin (A, C)$ and thus there exists a x-variable x_i such that the value of x_i for A and C is equal to say a and the value of x_i for B is a'. $B \in (A, D)$ implies the value of the x_i variable for D equals a'. But $C \in (B, D)$, implies the value of x_i for C is a'; hence, a contradiction. Therefore, a static function hazard exists for input changes A to C and B to D. Because $f(A) = f(C) \neq f(B) = f(D)$ implies $f(A) = f(C) = 1$ and $f(B) = f(D) = 0$ or vice versa. Thus, the theorem.

A Boolean function f which is free of function hazards for input change A to B may still contain hazards due to delays in the logic gate realization for input change A to B. The next section will deal with elimination of hazards due to delays.

III. Logic Hazards

The following definitions will be used for elimination of hazards due to delays. Definition 4 is equivalent to the logic-hazard definition of Eichelberger (1964).

Definition 4. A network contains a *static logic hazard* for the input change A to B if and only if

(1) $f(A) = f(B)$.

(2) No static function hazard exists for input change A to B.

(3) During the input change A to B, a momentary pulse may be present on the output.

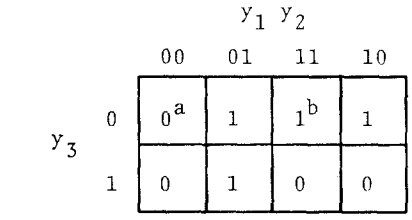
Definition 5. A network contains a *dynamic logic hazard* for the input change A to B if and only if

(1) $f(A) \neq f(B)$.

(2) No dynamic function hazard exists for input change A to B.

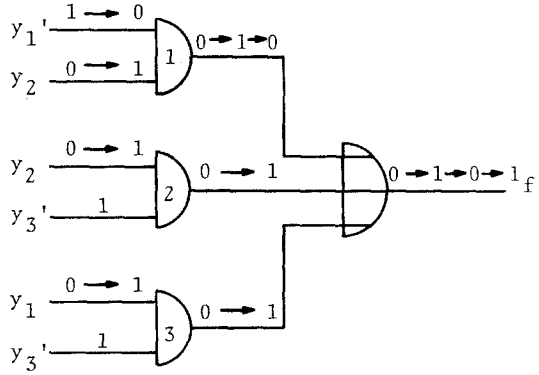
(3) During the input change A to B a momentary "0" output and a momentary "1" output may appear.

Eichelberger (1964) has shown that any sum-of-products realization of a function will be free of (static) logic hazards if and only if the realization contains all prime implicants. The same is not true for a dynamic logic hazard as shown by the following example.



$$f(y_1,y_2,y_3) = y_1'y_2 + y_2y_3' + y_1y_3'$$

(a) Map of Function



(b) Circuit Realization

FIG. 2. Example of a network containing a dynamic logic hazard but no static logic hazard.

Consider the function f as given in Fig. 2. Let us examine the transition from cell a to cell b. Assume y_2 changes before y_1 and gate 1 is significantly faster than gates 2 and 3. Then it is possible for the output of gate 1 to momentarily assume an output of "1" then "0" before either the output of

gate 2 or 3 changes to "1". Hence, a "0–1–0–1" transition on the output is possible. This sum-of-products realization contains all prime implicants and no dynamic function hazard in the transition from cell *a* to cell *b*. Thus the inclusion of all prime implicants will not suffice for elimination of dynamic logic hazards even though static logic hazards are eliminated. Theorem 4 defines the conditions when a sum-of-products realization of functions will not contain a dynamic logic hazard.

THEOREM 4. *Let f be a function which contains no dynamic function hazard for input change A to C, where $f(A) = 0$, $f(C) = 1$, and let B be any input such that $B \in (A, C)$ and $f(B) = 1$. A sum-of-products realization of f (assuming no product terms with complementary literals) will contain no dynamic logic hazard for input transition A to C (or C to A) if and only if for any B which is covered by a prime implicant α, then it is also true that α covers C.*

Proof. Assume each prime implicant which covers $B \in (A, C)$ also covers C. If we can show that each AND gate corresponding to a prime implicant will never change from "0–1–0" for input transition A to C, then obviously, f will contain no dynamic logic hazards. An AND gate output which corresponds to a prime implicant covering B will change to "1" only when an input state B is reached where $f(B) = f(C) = 1$ and $B \in (A, C)$. Since the prime implicant corresponding to the AND gate covers both B and C, it will not be a function of the literals in which B and C differ (a prime implicant can only be a function of the literals which are equal). Hence, any input $D \in (B, C)$ will produce a "1" on the output of any AND gate. Thus a 1 to 0 transition can never occur on the output of any AND gate.

Assume there exists a prime implicant which covers B and not C. The AND gate corresponding to this prime implicant will become "1" when the input state B is reached. Since the prime implicant does not cover C, the

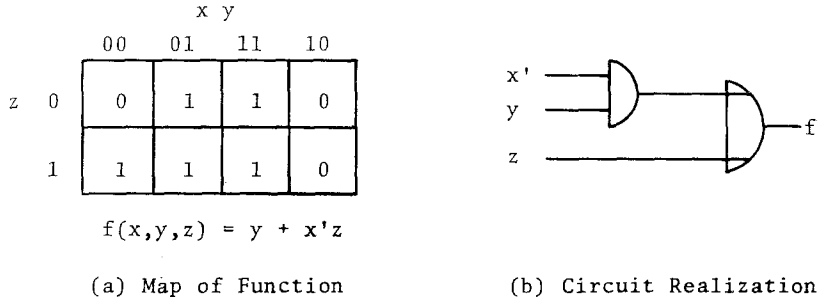


FIG. 3. Example of a network containing no logic hazards.

AND gate output must necessarily return to "0" when input C is reached. If this AND gate is faster than the remaining AND gates, it is obviously possible for a "0–1–0–1" transition to occur on the output of the OR gate. The case for input transition C to A can be similarly proved. Hence, the theorem.

Obviously a product-of-sums realization would have an equivalent dual theorem. A circuit which contains no logic hazards for any transition is given in Fig. 3.

IV. Elimination of Static and Dynamic Logic Hazards

The following method for realizing a Boolean function $f(x_1, x_2, \ldots, x_n)$ utilizing an SR flip-flop can be shown to produce a static and dynamic logic hazard free circuit. Let

$$S = y'f(x_1, x_2, \ldots, x_n), \tag{1}$$

$$R = yf'(x_1, x_2, \ldots, x_n), \tag{2}$$

where y is the output of the SR flip-flop. Realize both S and R with AND/OR logic using NOR–AND pair factoring for each AND gate to insure the x changes are felt before y changes. This method was first proposed by Armstrong, Friedman, and Menon (1968) which assumes line delay is less than minimum loop delay. The characteristic equation for an SR flip-flop is $Y = S + R'y$. Thus, for our case

$$\begin{aligned} Y &= y'f(x_1, x_2, \ldots, x_n) + [yf'(x_1, x_2, \ldots, x_n)]'y \\ &= y'f(x_1, x_2, \ldots, x_n) + [y' + f(x_1, x_2, \ldots, x_n)]y \\ &= y'f(x_1, x_2, \ldots, x_n) + yf(x_1, x_2, \ldots, x_n) \\ &= f(x_1, x_2, \ldots, x_n). \end{aligned}$$

The Y output of the SR flip-flop is $f(x_1, x_2, \ldots, x_n)$. The following theorem insures that no logic hazards occur on the Y output. Y is the next value y assumes.

Theorem 5. *Any function $f(x_1, x_2, \ldots, x_n)$ realized with a circuit in the above manner with an SR flip-flop will contain no static or dynamic logic hazards on the $Y = f(x_1, x_2, \ldots, x_n)$ output when line delay is less than loop delay.*

Proof. First we will show no static logic hazards exist for input change A to B. Assume $f(A) = f(B) = 0$. If a static logic hazard existed, then no

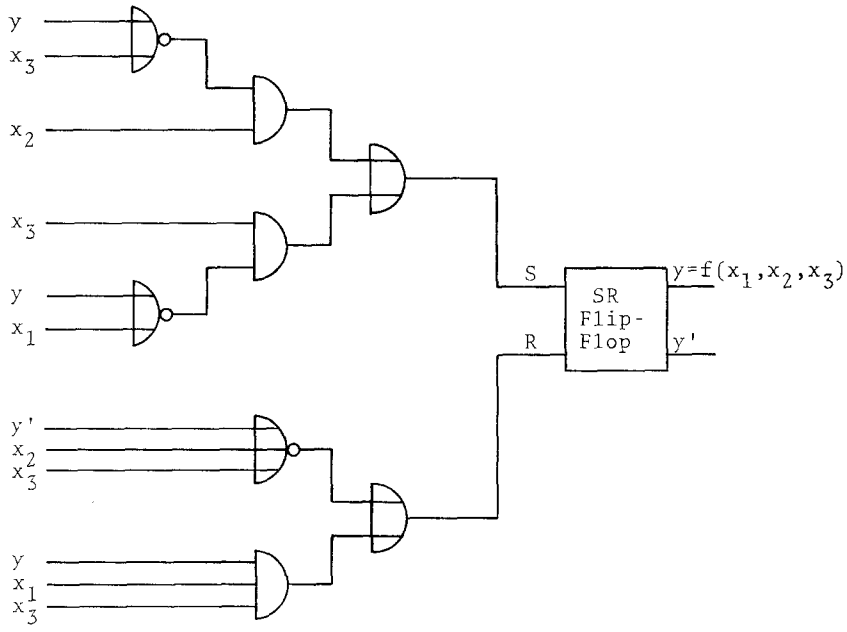


FIG. 4. Example of logic hazard-free circuit using *SR* flip-flops for $f(x_1, x_2, x_3) = \Sigma(1, 2, 3, 6)$.

static function hazard can exist, i.e., there exists no $C \in (A, B)$ such that $f(C) = 1$. Thus a two-level AND/OR network realizing $S = y'f(x_1, x_2, ..., x_n)$ must have inputs which always produce zero at the output. The absence of complementary literals within a product term eliminates the possibility of a "0–1–0" static logic pulse occurring on the S input. Since $S = 0$ for the entire transition, the Y output will always remain at 0. A similar argument holds for $f(A) = f(B) = 1$; hence, no static logic hazards will exist.

Now we will show no dynamic logic hazards exist for input change A to B where $f(A) = 0$ and $f(B) = 1$. The final state of the *SR* flip-flop will obviously be "1". If we can show the R input remains at "0" for the entire input change, then obviously a dynamic static hazard cannot exist on output Y. If the transition could contain a dynamic logic hazard, then it cannot contain a dynamic function hazard. All input combinations into the two level AND/OR network will be shown to give a "0" output. When $y = 0$, obviously $R = yf'(x_1, ..., x_n) = 0$. The logic value of y will never equal "1" until $S = 1$. This implies there exists an input state C such that $f(C) = 1$, where $C \in (A, B)$. Thus, $f'(C) = 0$, which implies $R = 0$. All input states $D \in (C, B)$

must give $f(D) = 1$ or $f'(D) = 0$ (by Theorem 2). Thus, $f'(x_1, x_2, ..., x_n)$ becomes "0" and remains at "0" before y changes to "1". Hence, the inputs into the R network for the input transition A to B will always produce a zero output. If NOR–AND pairs are used for all product terms to allow the network to see the x changes before y changes (assume no complementary literals within a product term), then none of the AND gates will be "1". Hence, no hazard pulses will be present on the R input. A similar argument holds for $f(A) = 1$ and $f(B) = 0$. Hence the theorem.

In the above proof, we assume that if the flip-flop is in the set state ($y = 1$), then any "1" pulse on the R input which is of sufficient duration to change y from 1 to 0 will insure that y' will go from $0 \rightarrow 1$. This forces y to remain at zero even when R disappears. We also assume that the same condition is met when a "1" pulse occurs on S if the flip-flop is in the reset state. This may not be the case for cross-coupled NOR gates in Fig. 5. It is possible if one NOR gate is faster than the other for a momentary "1" pulse on R to cause a $1 \rightarrow 0 \rightarrow 1$ transition on y. This situation can be remedied as follows if the RS flip-flop is constructed as in Fig. 6. It should be apparent that this is an RS flip-flop with $z = y$ and $z' = y'$ under steady

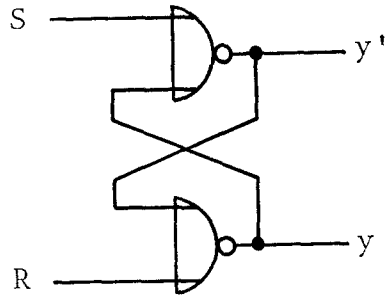


FIG. 5. RS flip-flop with two cross-coupled NOR gates.

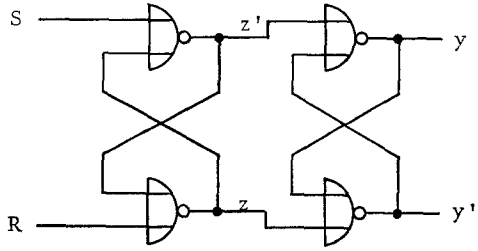


FIG. 6. RS flip-flop with four NOR gates.

state conditions. Let us examine the flip-flop action when $S = 0, y = z = 1$, and $y' = z' = 0$. For cross-coupled NOR gates it is impossible for $zz' = 1$ under transient conditions. This implies transients may occur on z only when $z' = 0$. These transients have no effect on y and y', since z' becomes "1" only when $z = 0$. Since z' cannot go to "0" ($S = 0$), this implies z' and z will remain at "1" and "0" respectively. Thus z' will have only a $0 \rightarrow 1$ transition which implies y and y' can have only $1 \rightarrow 0$ and $0 \rightarrow 1$ transition respectively. It should be apparent that multiple transitions will not occur on y and y' when $R = 0$ and pulses of arbitrary length appear on S. Thus this flip-flop will have no multiple transitions on the outputs y and y'.

An example of the procedure is given for $f(x_1, x_2, x_3) = \Sigma(1, 2, 3, 6)$. The inputs to the SR flip-flops are

$$S = y'x_2x_3' + y'x_1'x_3, \qquad R = yx_2'x_3' + yx_1x_3.$$

To insure changes in x are seen before changes in y NOR–AND pair factoring is used to give

$$S = x_2(y + x_3)' + x_3(y + x_1)', \qquad R = (y' + x_2 + x_3)' + yx_1x_3.$$

The circuit is given in Fig. 4.

The AND–OR circuit realization of the S and R inputs to the flip-flop requires no special factoring. Thus, only minimal sum-of-products realization is necessary. If the proposed method is used to eliminate only static logic hazards, then NOR–AND pair factoring is not needed to insure the x changes are felt before the y changes (y does not change for the static case). In some cases, less total logic is needed for our method than Eichelberger's (1964) where all prime implicants or prime implicates must be used.

Consider the following function

$$f(x_1, x_2, x_3, x_4, x_5, x_6) = \sum (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 20, 27, 39, 40, 48, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 63).$$

The number of AND gates, number of OR gates and inputs are listed for each method.

Eichelberger's (Sum of Products)		Eichelberger's (Product of Sums)	
24	– 4 input AND	24	– 4 input OR
8	– 5 input AND	8	– 5 input OR
1	– 32 input OR	1	– 32 input AND
Total: 33 gates – 168 inputs		Total: 33 gates – 168 inputs	

SR Flip-Flop

12 – 5 input AND
8 – 6 input AND
2 – 10 input OR
2 – 2 input NOR
(for *SR* flip-flop)

Total: 24 gates – 132 inputs

V. Conclusion

Two types of hazards have been defined. The first, function hazards, are a characteristic of the Boolean function and cannot be realized with circuitry to insure that no spurious pulses will be present on the output. The second, logic hazards, are pulses due to the delays in the logic. Eichelberger (1964) examined the function and logic hazards when the output remained fixed (static case) during the input transition. We have examined the case when the output is to change (dynamic case) during the input transition.

We have shown the interrelation between static and dynamic function hazards and have given an implementation scheme to eliminate any static or dynamic logic hazards under the assumption line delay is less than loop delay.

Received: September 30, 1970; Revised: November 22, 1971

References

Armstrong, D. B. (1968), Realization of asynchronous sequential circuits without inserted delay elements, *IEEE Trans. Computers* **C-17**, 129–134.

Eichelberger, E. B. (1964), Hazard detection in combinational and sequential switching circuits, *IEEE Int. Conf. Switching Circuit Theory Logical Design* **12**, 111–121.

Huffman, D. A. (1957), The design and use of hazard-free switching networks, *J. Assoc. Comput. Mach.* **4**. Jan. 1957, pp. 47–62.

McCluskey, E. J., Jr. (1962), "Redundancy Techniques for Computing Systems," Spartan Books, Washington, D.C., pp. 9–46.